# F-HB+: A Scalable Authentication Protocol for Low-Cost RFID Systems

# 13

# F-HB$^+$: A Scalable Authentication Protocol for Low-Cost RFID Systems

Xiaolin Cao and Máire P. O'Neill
*Centre for Secure Information Technology (CSIT),*
*Queen's University Belfast,*
*Northern Ireland*

## 1. Introduction

RFID technology has received much attention both in industry and academia in recent years and it is seen as the leading ubiquitous computing technology. A typical RFID system consists of a reader, $\mathcal{R}$, and a set of tags, $(\mathcal{T}_i)_{1\leq i\leq N}$. The reader $\mathcal{R}$ is composed of a set of transceivers and a powerful backend database. Each tag $\mathcal{T}_i$ is a passive transponder identified by a unique ID. However, the fact that RFID tags can be read without line-of-sight results in security risks, especially in relation to the privacy of tag users. Therefore, developing privacy preserving authentication protocols for low-cost RFID tags is a major security challenge that needs to be addressed if RFID systems are to be widely deployed in the coming years.

In previous research in this area the majority of authentication protocols use challenge-response mutual authentication based on symmetric-key ciphers. In order to preserve privacy, on receiving a challenge from a reader, a tag uses pseudonyms, which are the result of using symmetric-key ciphers to process the secret key or ID, as authenticators to the reader. The reason symmetric-key ciphers are used is that the hardware cost of existing asymmetric-key ciphers is too expensive for low-cost tags. For example, ECC and RSA require more than 40,000 gates, which are too large for low-cost tags in which only 200 – 2,000 gates out of 1,000 – 10,000 gates are available for security features (Juels, 2006). A lightweight algorithm known as the learning parity with noise (LPN) problem was first introduced in the HB protocol for human authentication by Hopper & Blum (2001). Juels & Weis (2005) first employed the LPN problem in the HB$^+$ protocol for RFID authentication. The simplicity and novelty of the HB$^+$ protocol has led to the proposal of other HB-related protocols (Jr *et al*., 2010). Gilbert *et al*. (2008) introduced a simple but effective man-in-the-middle attack against these types of protocols, in which the adversary can derive the secret key of the LPN problem through modifying the tag's response messages. This attack is known as a GRS-MIM attack. In the Trusted-HB protocol (Bringer & Chabanne, 2008), a universal hashing based message authentication code (MAC) is introduced to effectively resist GRS-MIM attacks. Although cryptographic attacks to the Trusted-HB protocol have been reported, they are impractical as they are too complex to implement (Frumkin & Shamir, 2009). Meanwhile, in the F-HB protocol (Cao & O'Neill, 2011), the LNP problem is first introduced to protect the forward privacy of low-cost tags. The operations in the LPN problem involve the calculation of inner products of binary vectors and Bernoulli noise bit

generation. Computing the binary inner product only requires bitwise AND and OR operations that can be computed on the fly. Therefore the LPN problem is a hardware-friendly primitive, and very attractive for low-cost RFID security. The most recent progress in LPN-based protocols was reported by Kiltz *et al.* (2011). They introduced an authentication protocol based on a variant of the LPN problem, known as the Subspace LPN problem, and also proposed an efficient MAC construction based on the LPN problem.

The rigorous definition and modelling of privacy in RFID systems has also been investigated in previous research (Avoine, 2005; Juels & Weis, 2007; Vaudenay, 2007; Ha *et al.*, 2008). This research differs in how they treat the adversary's ability to corrupt tags and their different privacy notions for corrupted tags. Compared to the general privacy notion that only considers adversaries that are unable to corrupt a tag, forward privacy is a stronger privacy notion because it also considers the privacy of a corrupted tag. Ma *et al.* (2009) prove that the unpredictable privacy notion (Ha *et al.*, 2008) is stronger than the indistinguishable privacy notion (Juels & Weis, 2007), and that the unpredictable privacy notion is equivalent to a pseudo random function (PRF). It can be observed that the majority of existing forward privacy schemes (Ohkubo *et al.*, 2003; Berbain *et al.*, 2009; Billet *et al.*, 2010) are based on the indistinguishable privacy notion, and the F-HB protocol is based on the unpredictable privacy notion.

Scalability must also be considered in forward private protocols based on symmetric-key ciphers. In order to protect a tag's privacy, before the tag is authenticated by the reader, it must not reveal its identity (its secret key) to the reader. As a result, in order to locate the identity of a tag, the reader must perform a brute-force search of all the tags to check all the keys in its database. As the number of tags increases, this brute-force search will inevitably lead to scalability problems. Existing research into scalability protocols are composed of three categories. The first category comprises protocols that perform a brute-force search of all the tags in the database (Weis *et al.*, 2003; Ohkubo *et al.*, 2003), the time complexity of which is $O(N)$, where $N$ is the number of tags in the system. This method is only suitable for systems with a small number of tags. The second category involves tree-based protocols (Molnar and Wagner, 2004; Molnar *et al.*, 2005), with a time complexity of $O(\log_b N)$ where $b$ represents the branch factor of the tree. These protocols consider each tag as a leaf in a balanced tree, and each tag needs to store $\lceil \log_b N \rceil$ secrets corresponding to the path from the root to the tag leaf. The disadvantage of this method is that because this approach requires that each tag stores correlated keys, the system privacy is weakened when an adversary is able to corrupt at least one tag. The more tags that are corrupted, the more the privacy of this system is compromised. The advantage of this method is that it supports dynamic scalability, so that new tag entries can be easily added without affecting the operation of the protocol. The third category of scalable protocols are hash-table based protocols (Henrici and Muller, 2004; Dimitriou, 2005; Tsudik, 2006; Lim and Kwon, 2006; Le *et al.*, 2007; Song, 2009; Alomair *et al.*, 2010; Cao & O'Neill, 2011). These protocols require only constant-time, $O(1)$, running time to identify a tag. These protocols need to store pre-computed hash-tables in the database associated with the reader. The reader uses pseudonyms from a tag as the indices of the hash-table to match a value, realizing constant-time tag identification. Compared to the tree-based protocols, hash-table based protocols need smaller storage on a tag and maintain a constant response time even when the number of tags increases. The disadvantage of these protocols is that the backend database needs a large storage to build a hash-table. Although, it is assumed that in RFID systems the database possesses infinite computational ability, from a practical viewpoint, all previously proposed protocols in this category require unrealistic large storage, and lack dynamic scalability (Avoine *et al.*, 2010).

In this chapter, building on previous work in this area, a novel scalable and forward private authentication protocol, F-HB+, suitable for low-cost RFID applications is proposed. The contributions are as follows. Firstly, similar to the F-HB protocol, the proposed protocol uses an LPN problem and a pseudo random number generator (PRNG); however, a hardware counter is introduced to the tag to enhance its desynchronization resistance, and the MAC code generation based on the proposal of Kiltz *et al.* (2011) is more efficient than in the F-HB protocol. Secondly, a new Re-Hash technique is presented to effectively reduce the storage requirement of the hash-table over previous protocols. The Re-Hash technique is adapted to support dynamic scalability and it is used to construct the hash-table required in the F-HB+ protocol. Thirdly, the security proof of the F-HB+ protocol is derived under the standard model. Overall, the proposed protocol features: (i) from the tag's perspective, low-cost implementation and forward privacy; (ii) from the reader's perspective, constant-time scalability, small hash-table storage and dynamic scalability.
The rest of the chapter is organized as follows. In section 2, the mathematical definitions and previous related work are introduced. In section 3, the Re-Hash technique is presented, and how it can be adapted to include dynamic scalability is discussed. The proposed F-HB+ scheme with the Re-Hash technique is described in section 4. The unpredictable forward privacy framework and security proof are derived in section 5. Section 6 presents a performance evaluation and comparison results, while Section 7 concludes the chapter.

## 2. Preliminary

### 2.1 Mathematical definitions

**Definition 1.** LPN Problem (Hopper & Blum, 2001). Let $\mathrm{Ber}_\eta$ denote the Bernoulli distribution with parameter $\eta \in (0,1/2)$. A bit $v \leftarrow \mathrm{Ber}_\eta$ is such that $\Pr[v = 1] = \eta$ and $\Pr[v = 0] = 1 - \eta$, while an $l$-bit vector $v \leftarrow \mathrm{Ber}_{l,\eta}$ is such that each bit of $v$ is independently drawn according to $\mathrm{Ber}_\eta$. Let $\mathrm{Hwt}(v)$ denote the hamming weight of vector $v$. Let $T$ be a random $(l \times n)$ binary matrix, let $x$ be a random $n$-bit vector, let $\eta \in (0,1/2)$ be a noise parameter, and let $v$ be a random $l$-bit vector according to $\mathrm{Ber}_{l,\eta}$, such that $\mathrm{Hwt}(v) \leq \eta l$. Given $T$, $\eta$ and $z = (T \cdot x) \oplus v$, find an $n$-bit vector $y$ such that $\mathrm{Hwt}((T \cdot y) \oplus z) \leq \eta l$.
For a fixed $n$-bit string, $k$, let $\pi_{k,\eta}$ denote the oracle returning an independent $(n+1)$-bit string according to the LPN problem:

$$\{(a, (k \cdot a) \oplus v) | a \in_R \{0,1\}^n, v \leftarrow \mathrm{Ber}_\eta\}. \quad (1)$$

The following Lemma 1 upper-bounds the probability that an adversary predicts the secret $n$-bit string $k$ given some instances of oracle $\pi_{k,\eta}$, which implies that the two oracles, $\pi_{k,\eta}$ and $U_{n+1}$, are computationally indistinguishable, where $U_{n+1}$ denotes an oracle that returns an independent uniformly random $n+1$-bit string.
**Lemma 1.** Indistinguishability of LPN Problem (Katz & Shin, 2006). Assume there exists an algorithm $A$ making $q$ oracle queries, running in time $t$, and such that

$$|\Pr[A^{\pi_{k,\eta}}(1^n) = 1] - \Pr[A^{U_{n+1}}(1^n) = 1]| \geq \epsilon. \quad (2)$$

Then there is an algorithm $B$ making $O(q \cdot \epsilon^{-2} \log n)$ oracle queries, running in time $O(t \cdot n\epsilon^{-2} \log n)$, and such that

$$\Pr[B^{\pi_{k,\eta}}(1^n) = k | k \in_R \{0,1\}^n] \geq \epsilon/4. \quad (3)$$

**Definition 2.** PRNG (Goldreich, 2001). A PRNG is a function $g: \{0,1\}^m \to \{0,1\}^n$ that takes as input an $m$-bit hidden seed and returns an $n$-bit string, where $n > m$. The output of the PRNG is called a pseudo random number, which appears to be random. A $(t, \epsilon_g)$-secure PRNG represents that the output of this PRNG cannot be discriminated with a true random string in time $t$ with advantage at most $\epsilon_g$.

The PRNG can be implemented using stream ciphers such as those proposed in the STREAM project (Cid & Robshaw, 2009) and a secure stream cipher is seen as a PRF (Billet *et al.*, 2010).

**Definition 3.** Universal Hash Functions (Wegman & Carter, 1981). A family of functions $\{h_u: \{0,1\}^l \to \{0,1\}^m\}_{u \in U}$ is called a strongly universal hash family if $\forall x \in \{0,1\}^l$, $\forall y \in \{0,1\}^m$:

$$\Pr[h_u(x) = y] = 2^{-m}, \tag{4}$$

and $\forall x_1 \neq x_2 \in \{0,1\}^l$, $\forall y_1, y_2 \in \{0,1\}^m$:

$$\Pr[h_u(x_2) = y_2 \,\&\, h_u(x_1) = y_1] = 2^{-2m} \tag{5}$$

where any hash function is easily selected by $u \in U$.

An $(l \times m)$-bit Toeplitz matrix is a matrix for which the entries on every upper-left to lower-left diagonal have the same value. Since the diagonal values of a Toeplitz matrix are fixed, the entire matrix is specified by the top row and the first column. Thus a Toeplitz matrix can be stored in $(l + m - 1)$ bits rather than the $(l \times m)$ bits required for a truly random matrix. For any $(l + m - 1)$-bit vector $u$, let $T_u$ denote the Toeplitz matrix whose top row and first column are represented by $u$.

**Definition 4.** Toeplitz based Universal Hash Function (Krawczyk, 1994). Let $\{T_u\}_{u \in U}$ be the family of Toeplitz matrices where the $(l + m - 1)$-bit vector $u$ is chosen at random, and $z$ is a random $m$-bit vector. Then the following is a strongly universal hash function family:

$$\{h_u(x) = (T_u \cdot x) \oplus z: \{0,1\}^l \to \{0,1\}^m\}_{u \in U}. \tag{6}$$

Meanwhile, according to the property in (5), the Toeplitz based universal hash function is also a pairwise independent hash function (Naor & Reingold, 1997).

**Definition 5.** LPN based MAC (Kiltz *et al.*, 2011). Let $h_u: \{0,1\}^l \to \{0,1\}^m$ be a pairwise independent hash function, $\rho(\cdot)$ be a pairwise independent permutation on $\{0,1\}^{l \times n + n + w}$, $v \leftarrow \mathrm{Ber}_{n,\eta}$, $s_i \in_R \{0,1\}^l$, $r \in_R \{0,1\}^w$, and $T \in_R \{0,1\}^{l \times n}$. Given a secret key $\left(\{s_i\}_{(0 \le i \le m)}, h_u, \pi\right)$ and a message $x$, the LPN based MAC for the message, $x$, can be defined as:

$$\mathrm{MAC}_{(s,h,\pi)}(x) = \rho(T, T^{\mathrm{T}} \cdot s(y) \oplus v, r), \tag{7}$$

where $y = h_u(x, r)$ and $s(y) = s_0 \oplus_{i:y[i]=1} s_i (0 \le i \le m)$.

The verification steps of the LPN based MAC are as follows. Firstly, use $\rho^{-1}(\cdot)$ to obtain $(T, z, r)$; if $\mathrm{rank}(T) \neq n$, then reject. Secondly, use $h_u(x, r)$ to obtain $y$ and $s(y)$. Thirdly, if $\mathrm{Hwt}\left(z \oplus T^{\mathrm{T}} \cdot s(y)\right) \le n\left(\frac{1}{4} + \frac{\eta}{2}\right)$, accept the MAC, otherwise reject.

One disadvantage of this MAC is that if the standard pairwise independent permutation $\rho(x) = a \times x + b$ (where $a$ and $b$ are random strings) is used, the computation for the multiplier will be a bottleneck for the LPN based MAC (Kiltz *et al.*, 2011). But it can be observed that the function of $\rho(\cdot)$ prevents the adversary from directly choosing the input of a MAC. The protocol proposed in this chapter solves this limitation by using a simplified

pairwise independent permutation, $\rho(x) = x + b$, where $a = 1$. Another disadvantage is that the key $(\{s_i\}_{(0\le i\le m)}, h_u, \pi)$ requires a large storage cost. The proposed protocol solves this by using a PRNG that is able to generate successive random strings.

### 2.2 Related work

In this section, a brief introduction and analysis of previous research is presented. The most relevant work for comparison is the hash-table based scalable and forward private protocols. These protocols can be divided into two classes according to their methods for generating pseudonyms. In the remainder of the chapter, the word "pseudonyms" is taken to mean indices used to look up a hash-table.

In the first class of protocols, each tag stores a unique key, which can be used as the tag's authenticator to the reader. The pseudonyms are derived from this secret key, and the pseudonym update method on the tag depends on a one-way secure hash function without interference from the reader. In the first hash-table based protocol proposed by Weis *et al.* (2003), on any query from a reader, a tag always replies with the fixed pseudonym of its unique secret key. Therefore, it is vulnerable to tracking attacks and tag impersonation. In the protocols proposed by Henrici and Muller (2004) and Dimitriou (2005), the tag's response comprises a pseudonym and an authenticator. Due to the fixed pseudonym used between successful mutual authentications, these protocols fail to resist tag tracking. The protocols proposed by Lim and Kwon (2006) and Tsudik (2006) also use a response pair. But the pseudonyms in these protocols will recycle in a brute-force desynchronization attack, so they fail to provide forward privacy.

In the second class of protocols, each tag needs to store two secrets, where one secret is used as the tag's final authenticator key and the other one is used to generate the pseudonym chain. These protocols possess the advantage that pseudonyms are unrelated to the secret key, but they use more non-volatile memory on the tag. The O-FRAP protocol was proposed by Le *et al.,* (2007) for RFID authentication under a universally composable framework and provides forward privacy. It updates pseudonyms using the same method as in the first class of protocols. The O-FRAP protocol constructs a hash-table using the output of a PRF implemented by a PRNG. But it is difficult to validate that the output of a PRF possesses the collision-free property. Two further protocols in this class (Song, 2009; Alomair *et al.,* 2010) require the help of the reader to update pseudonyms and send the updated pseudonyms to tags, which does not relieve the burden on the tag and adds to the risk of desynchronization. The desynchronization threats in the above protocols can be alleviated by using more than one pseudonym for a secret key. There are two methods to achieve this purpose. One method is based on the time-stamp concept (Tsudik, 2006), and involves adding a hardware timer to the tag, inevitably increasing the cost of the tag. This technique is unsuitable for low-cost tags. Another technique relies on a hardware counter on the tag (Le *et al.,* 2007; Song, 2009; Alomair *et al.,* 2010). This counter is used to limit the maximum number of pseudonyms associated with a secret key. The maximum threshold value of this counter determines the ability to resist desynchronization attacks. Although the hardware counter also increases the cost of the tag, it is more practical than a hardware timer. Another problem of the above protocols is that they utilise cryptographic secure hash functions, the hardware cost of which exceeds the budget of low-cost tags. For example, according to the latest literature reports, the standard algorithm, SHA-1, requires at least 5,000 gates (O'Neill, 2008).

The most recent progress in constant-time scalable protocols is presented by Alomair *et al.* (2010). It also uses a counter with threshold $Th$ to control the number of pseudonyms for each secret key. Compared to the previous proposals, this protocol considers a further step: how to build a hash-table with a reasonable storage in the database. This paper points out that impractically large hash tables are a result of the fact that the bit-length of a pseudonym, $L$, must be long enough to avoid collision. And in order to directly address the hash-table, the size of the hash-table must be $O(2^L)$ bits, which is unrealistic in practice. In order to reduce the storage requirement, a 2-level hash-table construction method is proposed. The 1st level is a hash-table with the $s$ most significant bits (MSB) of the $L$-bit pseudonyms as its indices, and that stores the addresses of the 2nd level. The 2nd level is a linear table composed of the remainding $(L - s)$ bits of the $L$-bit pseudonym, that stores the addresses of the actual information. Assuming that the number of pseudonyms is $N'$, the protocol recommends the use of the following parameters: the 1st level storage is $O(2^s)$ bits, where $s = \lceil \log_2(N' \times Th) \rceil$, and the 2nd level storage is $O(N' \times Th)$ bits. Using these parameters, constant-time authentication can be achieved with the 2-level hash-table. Avoine *et al.* (2010) noted that although this method is very efficient, its total storage requirement for the 2-level structure is still very large and does not support dynamic resizing.

## 3. Proposed Re-Hash technique

### 3.1 Basic Re-Hash technique

As mentioned before, in the hash-table based protocols, a tag can be identified in constant-time by its $L$-bit pseudonyms. The total number of valid pseudonyms for each tag in a synchronized state is controlled by a counter with a maximum threshold, $Th$. Firstly, let us take an example to show how much storage is required if these pseudonyms are directly used as look-up indices of a hash-table. The total number of tags, $N$, is assumed to be $2^{30}$ (greater than 1 billion) and the value of $Th$ is $2^{10}$. Therefore $2^{40}$ $(= N \times Th)$ indices are needed for the hash-table, so the collision-free bit-length of an index should be at least 40 bits. According to Alomair *et al.* (2010), the bit-length of pseudonyms should be large enough to obtain a collision-free 40-bit index of a hash-table. Assuming $L = 60$ bits, the collision-free hash-table needs at least $2^{17}$ terabytes (TB) of storage with $2^{60}$ slots ($2^{60} \times 1$ bit, i.e., assume every slot in the hash-table stores 1 bit) to meet the demands of direct addressing. This storage requirement is too large for practical use.

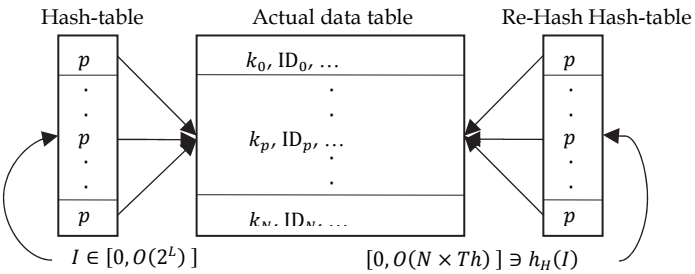


Fig. 1. The traditional Hash-table vs. basic Re-Hash hash-table

It can be observed that in the above example only $2^{40}$ slots out of the total $2^{60}$ slots are used in each authentication session, so that the truly useful storage of all the indices during each authentication session is 0.125 TB ($2^{40} \times 1$ bit), which is practical. Therefore, of the total $O(2^L)$ bits of storage, the true requirement is at most $O(N \times Th)$ bits, which causes a huge storage waste.

Therefore, in order to reduce the storage cost, a mathematical mapping is needed, $f: \{0,1\}^{60} \rightarrow \{0,1\}^{40}$, which is the essence of the Re-Hash technique proposed in this chapter. The function $f(\cdot)$ can be implemented as a look-up table hash function $h_H(\cdot)$, which uses the 60-bit pseudonyms of tags as its inputs and outputs 40-bit strings. These 40-bit outputs can then be used as look-up indices of a hash-table. If this technique is used, the storage cost of the directly addressed hash-table in the above example can be reduced to 0.125 TB ($2^{40} \times 1$ bit). Fig. 1 illustrates the difference between the traditional hash-table and the basic Re-Hash hash-table, where $I$ represents the pseudonym of a tag, and $p$ represents the address of the actual information related to the tag.

The Re-Hash technique for hash-table construction can be generalized as follows:

1. Determine the number of pseudonyms required during each authentication session, $N \times Th$, in the RFID system.
2. Determine the collision-free bit-length of a pseudonym, $L$.
3. Select an appropriate look-up table hash function, $h_H: \{0,1\}^L \rightarrow \{0,1\}^{N \times Th}$, which uses the pseudonyms as its input values.
4. Use the output of $h_H$ as indices to construct the hash-table, in which every slot stores a pointer to the address storing actual tag information.

The important advantage of this technique is the storage cost saving. One possible disadvantage is that the collision probability among hash-table indices may increase, because the number of hash-table indices is equal to the number of pseudonyms in each authentication session. However in section 6.1 analysis shows that if an appropriate Re-Hash hash function is used, constant-time look-up is maintained.

### 3.2 Dynamic Re-Hash

In this section it is illustrated that it is necessary to build a dynamic hash-table to accommodate frequent database changes, insertions and deletions. Firstly, dynamic table should effectively utilize the storage available. Assume a large-scale supermarket respectively sells and buys $2^{20}$ (greater than 1 million) items per month, the change in the number of indices for the hash-table is $2^{31}$ ($2 \times 2^{20} \times 2^{10}$). Thus, the change in storage will be at least 2 gigabytes (GB) ($2^{31} \times 1$ bit). If the hash-table is fixed, then this 2 GB storage may not be fully utilized. Secondly, a dynamic table should be able to process concurrent transactions without affecting the system response time. For example, merchandize is checked out in a supermarket at the same time. This would need many hash-table insertions and deletions at the same time.

Linear-Hashing (Black, 2009) is a dynamically updateable hash-table construction method which implements a hash-table that grows or shrinks one slot at a time through splitting a current slot into two slots. In general, assuming the Linear-Hashing scheme has an initial hash-table with $M$ slots, then it needs a family of look-up table hash functions $h_{H,j}(\cdot) = f(\cdot) \bmod (2^j M)$. At any time, there is a value $j(\geq 0)$ that indicates the current splitting round and the current look-up hash functions; a pointer $p \in [0, \ldots, 2^j M - 1]$ which points to the slot to be split next; a total of $(2^j M + \mathrm{p})$ slots, each of which consists of a primary page and

possibly some overflow pages; and two hash functions $h_{H,j}$ and $h_{H,j+1}$. The look-up process works as follows: If $h_{H,j}(\cdot) \geq p$, choose slot $h_{H,j}(\cdot)$ since this slot has not been split yet in the current round; otherwise, choose slot $h_{H,j+1}(\cdot)$, which can either be the slot $h_{H,j}(\cdot)$ or its split image slot $h_{H,j}(\cdot) + 2^j M$.

The final proposed dynamic hash-table construction method, in which the Re-Hash technique is adapted to include the Linear-Hashing technique, can be described as follows:

1. Determine the system capacity, i.e., the maximum tag number $N_{MAX}$ the system can accommodate, and the collision-free bit-length of a pseudonym $L$.
2. Determine the output range of the Re-Hash hash function, $L'$, such that $L' \geq L/2$.
3. Select an appropriate look-up table hash function, which is used as the Re-Hash hash function, $h_H: \{0,1\}^L \rightarrow \{0,1\}^{L'}$.
4. Determine the initial tag number of this RFID system, $N$, and the initial dynamic hash-table size, $M$, such that $M \geq N \times Th$.
5. Determine the Linear-Hashing look-up hash function family, $h_{H,j}(\cdot) = h_H(\cdot) \bmod (2^j M)$.
6. Use the outputs of $h_{H,j}(\cdot)$ as indices to construct the dynamic hash-table, in which every slot stores a pointer to the address storing actual tag information.

## 4. F-HB$^+$ protocol description

### 4.1 Initialization

The initialization steps involved in the proposed F-HB+ protocol are as follows.

- Tag: Every tag is independently assigned a secret key $k \in_R \{0,1\}^m$, which is shared with the reader. Each tag can compute a PRNG $g(\cdot)$ as in Definition 2, multiple instances of $\pi_{k,\eta}$ at the same time, and an $m$-bit counter $ct_{\mathcal{T}} \leftarrow 0$ whose maximum threshold value is $Th$. They also have enough non-volatile memory to store the value of $k$ and $ct_{\mathcal{T}}$.
- Reader: In the database, there is an old key $k_{old} \leftarrow k$, a current key $k_{cur} \leftarrow k$, a counter $ct_{\mathcal{R}} \leftarrow 0$ with threshold $Th$, and $Th$ hash-table entries $\{h_{H,j}(I_i) \mid 0 \leq \mathrm{i} < Th\}$ for every tag, where $I_i = (T_k \cdot i) \oplus r_i$ and $r_i$ is the $i$-th iteration result of $g(k_{cur})$. The two secret keys are used to resist brute-force desynchronization attacks, and the $Th$ hash-table entries are used to enhance the desynchronization resistance. The variables for Linear Hashing are also initialized: the current splitting round indicator $j \leftarrow 0$ and the current splitting pointer $p_s \leftarrow 0$. All the information is organized into a pre-computed 2-level database structure, which is illustrated in Fig. 2. In addition, the database can compute a look-up hash function family $\{h_{H,j}(\cdot)\}_{j \geq 0}$. The 1st level of the database is the pre-computed

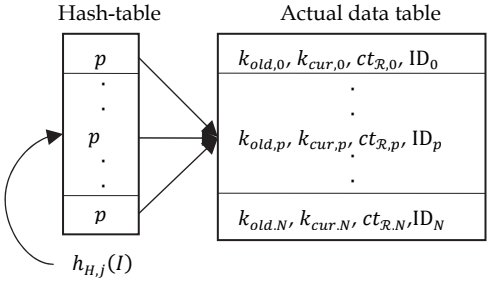


Fig. 2. The 2-level Database Structure with a Re-Hash Hash-table

dynamic hash-table. For every tag, there are $Th$ slots (maybe not successive) in this hash-table, which store the pointers $p$ indicating an address in the 2nd level table. The address of the 1st level hash-table is computed by $h_{H,j}(I_i)$. The 2nd level of the database is a pre-organized linear table. For each tag, there is only 1 slot in this level to store $k_{old}$, $k_{cur}$, $ct_{\mathcal{R}}$ and the actual information about each tag.

### 4.2 Authentication interaction

An overview of the proposed authentication protocol is illustrated in Fig. 3. It is a 3-pass mutual authentication protocol.

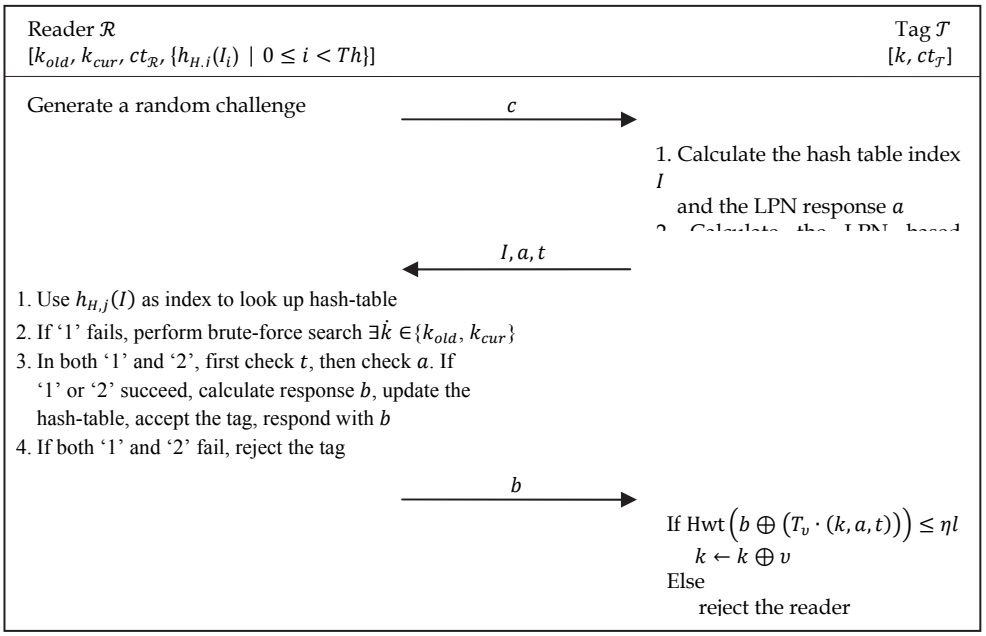


Fig. 3. The Proposed F-HB+ Protocol

Fig. 4 illustrates the tag's operation after the tag receives the challenge message $c$ from the reader. It can be observed that the Toeplitz matrix $T_k$ is used in the LPN problem such that $a \leftarrow \left(T_k \cdot (c, I)\right) \oplus v$, and in the strong universal hashing such that $I \leftarrow (T_k \cdot ct_{\mathcal{T}}) \oplus r$ at the same time. Meanwhile, the PRNG $g$ is also used in the strong universal hashing such that $\{r \leftarrow g(k), I \leftarrow (T_k \cdot ct_{\mathcal{T}}) \oplus r\}$. More importantly, the PRNG is in charge of generating all the secret keys of the LPN based MAC, such that $\left(\{s_i\}_{(0 \le i \le m)}, r_1, r_2\right) \leftarrow g(k)$.

Fig. 5 explains the reader's key search method in detail after it receives the authentication message $(I, a, t)$ from the tag. Only if both the MAC code $t$ and authenticator $a$ pass the verification will the reader accept the tag and generates a confirmation message, $b$. It can be observed that the reader does not use $k_{cur}$ as the secret key for the LPN problem again, but uses the noise vector $v'$ such that $b \leftarrow \left(T_{v'} \cdot (k_{cur}, a, t)\right) \oplus v''$. This is to prevent GRS-MIM attackers from recovering the secret key $k_{cur}$. The difference between steps 1 and 2 is that (i) step 1 only involves the current key $k_{cur}$ of one tag providing constant-time

scalability; but (ii) step 2 involves the secret key pair $(k_{old}, k_{cur})$ of all the tags, and needs to try all keys.

| Step 1: | Step 2: |
|---|---|
| $v \leftarrow \text{Ber}_{l,\eta}$, $r \leftarrow g(k)$<br>If $ct_{\mathcal{T}} < Th$<br>$I \leftarrow (T_k \cdot ct_{\mathcal{T}}) \oplus r$, $ct_{\mathcal{T}} \leftarrow ct_{\mathcal{T}} + 1$<br>Else<br>$I \in_R \{0,1\}^n$, $ct_{\mathcal{T}} \leftarrow ct_{\mathcal{T}}$<br>$a \leftarrow \left(T_k \cdot (c, I)\right) \oplus v$ | Generate random $r$ and $T$, $v \leftarrow \text{Ber}_{n,\eta}$,<br>$\left(\{s_i\}_{(0\le i\le m)}, r_1, r_2\right) \leftarrow g(k)$,<br>$y \leftarrow \left(T \cdot (c, a, I, r)\right) \oplus r_1$,<br>$s(y) = s_0 \oplus_{i:y[i]=1} s_i (0 \le i \le m)$<br>$t = (T, T^{\mathrm{T}} \cdot s(y) \oplus v, r\ ) + r_2$, |

Fig. 4. Tag's response operation in the Proposed F-HB+ Protocol

| Step 1: | Step 2: |
|---|---|
| $r \leftarrow g(k_{cur})$, $\left(\{s_i\}_{(0\le i\le m)}, r_1, r_2\right) \leftarrow g(k_{cur})$<br>$(T, z, r\ ) \leftarrow t - r_2$, if rank$(T) \neq n$, reject<br>$y \leftarrow \left(T \cdot (c, a, I, r)\right) \oplus r_1$<br>$s(y) = s_0 \oplus_{i:y[i]=1} s_i (0 \le i \le m)$<br>If Hwt$\left(z \oplus T^{\mathrm{T}} \cdot s(y)\right) \le n\left(\frac{1}{4} + \frac{\eta}{2}\right)$<br>$v' \leftarrow \left(T_{k_{cur}} \cdot (c, I)\right) \oplus a$<br>If $ct_{\mathcal{R}} < Th$ and Hwt$(v') \le \eta l$<br>$v'' \leftarrow \text{Ber}_{l,\eta}$, $ct_{\mathcal{R}} \leftarrow 0$<br>$b \leftarrow \left(T_{v'} \cdot (k_{cur}, a, t)\right) \oplus v''$<br>$(k_{old}, k_{cur}) \leftarrow (k_{cur}, k_{cur} \oplus v')$<br>update $\{h_{H,j}(I_i) \mid 0 \le i < Th\}$<br>accept the tag | $r \leftarrow g(\dot{k})$, $\left(\{s_i\}_{(0\le i\le m)}, r_1, r_2\right) \leftarrow g(\dot{k})$<br>$(T, z, r\ ) \leftarrow t - r_2$, if rank$(T) \neq n$, reject<br>$y \leftarrow \left(T \cdot (c, a, I, r)\right) \oplus r_1$,<br>$s(y) = s_0 \oplus_{i:y[i]=1} s_i (0 \le i \le m)$<br>If Hwt$\left(z \oplus T^{\mathrm{T}} \cdot s(y)\right) \le n\left(\frac{1}{4} + \frac{\eta}{2}\right)$<br>$v' \leftarrow \left(T_{\dot{k}} \cdot (c, I)\right) \oplus a$<br>If $ct_{\mathcal{R}} < Th$ and Hwt$(v') \le \eta l$<br>$v'' \leftarrow \text{Ber}_{l,\eta}$, $ct_{\mathcal{R}} \leftarrow 0$<br>$b \leftarrow \left(T_{v'} \cdot (\dot{k}, a, t)\right) \oplus v''$<br>$(k_{old}, k_{cur}) \leftarrow \left(\dot{k}, \dot{k} \oplus v'\right)$<br>update $\{h_{H,j}(I_i) \mid 0 \le i < Th\}$<br>accept the tag |

Fig. 5. Reader's authentication operation in the Proposed F-HB+ Protocol

### 4.3 Hash-table update procedure

This protocol supports dynamic update. The update procedure consists of insertion and deletion. Let us first to describe the insertion procedure. There are two insertion scenarios. One is when a tag is successfully authenticated, the old secret key is updated for this tag, therefore, the associated old $Th$ pseudonyms also need to be updated. The other scenario is when new tags are added into the system, new pseudonyms should also be included. Assuming that there is a new pseudonym called $I_{new}$, and its corresponding hash-table index is $h_{H,j}(I_{new})$. Therefore, $I_{new}$ is inserted into the slot $h_{H,j}(I_{new})$ as follows:

- If no overflow occurs, its position is within the primary page of this slot. Insertion process is completed.
- Otherwise $I_{new}$ is put into the overflow page of the slot $h_{H,j}(I_{new})$. The pseudonyms in the current splitting slot $p_s$ are split into 2 slots: $p_s$ and $p_s + 2^j M$ using the look-up hash function $h_{H,j+1}(\cdot)$. The splitting pointer $p_s$ moves to the next slot, $p_s \leftarrow p_s + 1$. If $p_s \ge 2^j M$, increment the current splitting round indicator, $j \leftarrow j + 1$, and reset the splitting pointer, $p_s \leftarrow 0$. Insertion process is completed.

Deletion will cause the hash-table to shrink. Slots that have been split can be recombined. The operation of two slots merging together is the reverse of splitting a slot in the insertion process.

Overall, the update procedure can be divided into two stages. The first stage is to insert the new pseudonyms according to the above insertion procedure in an on-line mode, which runs concurrently with other transactions. The second stage is to delete the old pseudonyms according to the deletion procedure, which can be done in an off-line mode, in order to obtain optimal system performance.

## 5. RFID privacy definition and proof

### 5.1 Adversary assumptions

In this chapter, an adversary $A$ is assumed to be a probabilistic polynomial algorithm that is allowed to perform oracle queries during attacks. The reader side is assumed to be secure. The tag and wireless communication channel are assumed to be insecure, which means that an adversary can intercept all the wireless communications between the reader and tags, and can corrupt a tag. The reader is assumed to have the ability to handle several authentication exchanges simultaneously, but a tag cannot. In order to model the majority of known attacks against authentication protocols in RFID systems, five oracles are defined as follows.

i. $O_1(\mathcal{R})$: It invokes the reader $\mathcal{R}$ to start a new session of the authentication protocol. This oracle returns the reader's challenge message $c$.
ii. $O_2(\mathcal{T}_i, c)$: It invokes a tag $\mathcal{T}_i$ to start an authentication session exchange related to challenge message $c$. The tag $\mathcal{T}_i$ responds with the response message $a$.
iii. $O_3(\mathcal{T}_i, c, a)$: It returns the unmodified and modified challenge, $c$, and response, $a$, related to a tag $\mathcal{T}_i$.
iv. $O_4(\mathcal{T}_i)$: It returns the final authentication result of a tag $\mathcal{T}_i$.
v. $O_5(\mathcal{T}_i)$: It returns the current key and internal state information of a tag $\mathcal{T}_i$, and also updates the key and state information of tag $\mathcal{T}_i$ if necessary.

For example, eavesdropping can be modelled as: first query $O_1$ to get $c$, then query $O_2$ to get $a$, and finally query $O_4$ to get authentication results. The message interception can be modelled by $O_3$. Any key compromised due to tag corruption, or side-channel attacks can be modelled by sending the $O_5$ query to the tag.

**Definition 6.** $(q, t)$-adversary. An adversary whose running time is upper-bounded by $t$ and has the ability to disturb at most $q$ authentication exchanges in this interval is called a $(q, t)$-adversary. The adversaries are assumed to only be able to attack the RFID system at a specific position and during a limited time period. The term "exposure period" (Vaudenay, 2007) is used to name this specific attack time. During an exposure period, an adversary is able to observe and disturb all interactions involving a target tag $\mathcal{T}_i$ and a legitimate reader $\mathcal{R}$ using oracle $(O_i)_{1\le i\le 5}$ according to the defined security model. After an exposure period, no adversary is allowed to continue his attack. But attacks do not need to be completed within only one exposure period, and can continue in several successive or discrete exposure periods.

### 5.2 LPN problem characteristics

From the protocol description, it can be found that in every authentication session, the tag needs to calculate multiple instances of $\pi_{k,\eta}$ at the same time: the secret is a Toeplitz matrix rather than a vector, the noise is a vector rather than a single bit. The usage is the same as in the HB# protocol (Gilbert *et al.*, 2008), but HB# reduces its security proof based on the hardness of the LPN problem. In this chapter, the security proof is based on the computational indistinguishability of the two oracles, $\pi_{k,\eta}$ and $U_{n+1}$, in Lemma 1.

First of all, a new oracle returning multiple bits of $\pi_{k,\eta}$ at the same time is defined as follows. For a fixed $(l \times n)$ matrix $K$, let $\Pi_{K,\eta}$ be the oracle returning an independent $(n+l)$-bit string according to:

$$\{(a, (K \cdot a) \oplus v) | a \in_R \{0,1\}^n, v \leftarrow \mathrm{Ber}_{l,\eta}\}. \tag{8}$$

Theorem 1 below upper-bounds the probability that an adversary predicts the secret $(l \times n)$ matrix $K$ given some instances of oracle $\Pi_{K,\eta}$, so it implies that the two oracles, $\Pi_{K,\eta}$ and $U_{n+l}$, are computationally indistinguishable.

**Theorem 1.** Assume there exists an algorithm $A$ making $q$ oracle queries, running in time $t$, and such that

$$|\Pr[A^{\Pi_{K,\eta}}(1^n) = 1] - \Pr[A^{U_{n+l}}(1^n) = 1]| \geq \epsilon. \tag{9}$$

Let $t_\pi$ be the time taken to calculate a $\pi_{k,\eta}$ instance. Then there is an algorithm $B$ making $O(q)$ oracle queries, running in time $t + \frac{l(l-1)}{2} t_\pi$, and such that

$$|\Pr[B^{\pi_{k,\eta}}(1^n) = 1] - \Pr[B^{U_{n+1}}(1^n) = 1]| \geq \epsilon/l. \tag{10}$$

Proof. A hybrid argument technique is used to prove it. Let $K'$ denote a $(l-j) \times n$ binary matrix. Firstly, define the following hybrid distribution, $D_j$, with $j \in [0, l]$ as

$$\{(a, r, (K' \cdot a) \oplus v)\}, \tag{11}$$

where $a \in_R \{0,1\}^n$, $r \in_R \{0,1\}^j$ and $v \leftarrow \mathrm{Ber}_{l-j,\eta}$. Upon receiving an $(n+1)$-bit input, $B$ gerneates a random value, $j \in [0, l]$ to construct an $(n+l)$-bit input as $A$'s input. When $j < l$, it also needs to generate a random $(l-j) \times n$ binary matrix $K'$. It is clear that when $B$'s input complies with $U_{n+1}$, $j \in [1, l]$; when $B$'s input complies with $\pi_{k,\eta}$, then $j \in [0, l-1]$. The distribution of $D_l$ is the same as $U_{n+l}$, and $D_0$ the same as $\Pi_{K,\eta}$. And $B$ uses $A$'s outputs as its outputs. Thus

$$\begin{aligned} &|\Pr[B^{\pi_{k,\eta}}(1^n) = 1] - \Pr[B^{U_{n+1}}(1^n) = 1]| \\ &= \frac{1}{l} \left| \sum_{j=0}^{l-1} (A^{D_j}(1^n) = 1) - \sum_{j=1}^{l} (A^{D_j}(1^n) = 1) \right| \\ &= \frac{1}{l} |\Pr[A^{\Pi_{K,\eta}}(1^n) = 1] - \Pr[A^{U_{n+l}}(1^n) = 1]| \geq \frac{\epsilon}{l}. \end{aligned} \tag{12}$$

A contradiction with the Lemma 1 is obtained, which concludes the proof.

**Defintion 7.** Indistinguishability of Oracle $\Pi_{K,\eta}$. The oracle $\Pi_{K,\eta}$ is said to be $(q, t, \epsilon)$-secure if there is no $(q, t)$-adversary who can distinguish $\Pi_{K,\eta}$ from $U_{n+l}$ with advantage $\epsilon$.

Secondly, due to the fact that Bernoulli random noise may exceed the acceptable threshold, even the legitimate tag may be rejected, which is called a false rejection. This property can also result in an adversary impersonating a tag successfully by simply guessing without any prior knowledge, which is called a false acceptance. According to probability theory, the false rejection probability $P_{FR}$, and false acceptance probability $P_{FA}$ in every authentication session can be defined as follows:

$$P_{FR} = \sum_{i=\eta l+1}^{l} \binom{l}{i} \eta^i (1-\eta)^{l-i}, \tag{13}$$

$$P_{FA} = \sum_{i=0}^{\eta l} \binom{l}{i} 2^{-l}. \quad (14)$$

Thirdly, in the protocol, the universal hashing MAC code is used to protect the integrity of communication messages. If the adversary uses the GRS-MIM attack and its variants (Gilbert *et al.*, 2008), the check for the universal hashing MAC code will fail, then, the reader will not continue to check the LPN problem as illustrated in Fig. 3. Therefore, the adversary cannot know whether or not his modification is successful according to the authentication result and the GRS-MIM attacks cannot succeed. Therefore, the GRS-MIM attack and its variants will not be considered in the following analysis.

### 5.3 Security

Experiment $\mathrm{Exp}_A^{\mathrm{Secure}}(\kappa, N, q, t)$
1. Setup a reader $\mathcal{R}$ and a set of tags $\mathcal{T}, |\mathcal{T}| = N$
2. $(\mathcal{T}_c, st_0) \leftarrow A^{(O_i)_{1\le i\le 5}}(\mathcal{R}, \mathcal{T})$ //learning stage, $q$ sessions
3. $A(\mathcal{R}, st)$ //guessing phase

Fig. 6. Security Experiment

An RFID authentication protocol is said to be secure if it resists impersonation attacks by any $(q, t)$-adversary without using relay or corruption attacks. Consider the experiment in Fig. 6. This experiment proceeds in two phases: a learning phase and a guessing phase. In the learning phase, the adversary $A$ is given an RFID system $(\mathcal{R}, \mathcal{T})$ as input. During a time interval at most $t$, $A$ is allowed to launch $(O_i)_{1\le i\le 5}$ oracle queries in every authentication session without exceeding $q$ sessions. At the guessing phase, adversary $A$ only interacts with the reader, and uses the information obtained from the learning phase to impersonate the tag $\mathcal{T}_c$, but can no longer access any oracle. Therefore, the security of an authentication protocol is defined as the successful impersonation probability in the above experiment.

**Theorem 2.** Let the oracle $\Pi_{K,\eta}$ in the F-HB⁺ protocol be $(q, t, \epsilon_\Pi)$-secure. Under the attack of a $(q, t)$-adversary, the security adversary's advantage of F-HB⁺ protocol is upper-bounded by:

$$\epsilon_s = P_{FA} + \frac{\epsilon_\Pi}{4l}. \quad (15)$$

Proof. The adversary may use two methods to impersonate a tag: (i) randomly guessing, and (ii) recovering the secret key (Toeplitz matrix). The successful probability of randomly guessing a response is $P_{FA}$ as mentioned before. Let us start to analyse how the adversary can deduce the secret key. There are two ways to obtain useful information about the tag's current key.

The first way is to block the tag's response message, as a result, the tag authentication is unsuccessful, and the current key cannot be updated. So the adversary can obtain valid instances of oracle $\Pi_{K,\eta}$, which can help to reveal the current key. According to Lemma 1 and Theorem 1, the probability of inferring the current key successfully is upper-bounded by $\frac{\epsilon_\Pi}{4l}$.

The second way is to block the reader's acknowledge message, as a result, the tag cannot update its current key. So the adversary can obtain valid instances of oracle $\Pi_{K,\eta}$, which can help to reveal the current key. Once again, the probability of inferring the current key is successfully is upper-bounded by $\frac{\epsilon_\Pi}{4l}$.

It is impossible that the adversary can block the two messages in the same session, because the reader or tag will terminate the session if they do not receive the corresponding message. Therefore, combining the situations above, for a $(q, t)$-adversary, the security of F-HB can be expressed as $\epsilon_s \leq P_{FA} + \frac{\epsilon_\Pi}{4l}$. This completes the proof.

### 5.4 Correctness

An authentication protocol exchange involving a legitimate tag and a legitimate reader is said to be undisturbed if all messages sent by both parties are correctly transmitted, received and neither modified nor lost in either direction.

The correctness for RFID authentication protocols implies that the legitimate reader should always accept the legitimate tag for all undisturbed authentications between them. But it is observed that the undisturbed session may happen before or after an attack. Therefore the correctness of an authentication protocol is defined as the acceptable probability of an legitimate tag in an undisturbed authentication session, where the tag may have experienced an impersonation attack.

**Theorem 3.** Let the oracle $\Pi_{K,\eta}$ in F-HB$^+$ protocol be $(q, t, \epsilon_\Pi)$-secure. Under the attack of a $(q, t)$-adversary, the correctness of the F-HB$^+$ protocol is at least:

$$\epsilon_c = (1 - {\epsilon_s}^2)(1 - P_{FR}) + {\epsilon_s}^2 P_{FA}. \tag{16}$$

Proof. According to the flow of the F-HB$^+$ protocol, a reader only rejects a legitimate tag when the tag cannot answer the challenge with a correct response. The reasons are composed of (i) falsely rejecting a tag as mentioned before, and (ii) an adversary successfully impersonating a tag two times in succession such that both the old and current keys are updated, thus, this tag cannot be authenticated again.

In the first situation, the correctness is at most $(1 - P_{FR})$ for a legitimate tag due to the inherent property of Bernoulli random noise, whenever this tag is under a synchronized (look-up table search) or desynchronized (brute-force search) state.

In the second situation, the probability of occurrence is ${\epsilon_s}^2$. Once this situation becomes true, this tag cannot be authenticated like a legitimate tag. But it still could be falsely accepted. So the correctness is ${\epsilon_s}^2 P_{FA}$.

Combining the two rejection situations, the correctness probability can be represented as $\epsilon_c = (1 - {\epsilon_s}^2)(1 - P_{FR}) + {\epsilon_s}^2 P_{FA}$. This concludes the proof.

### 5.5 Forward privacy

The unpredictable forward privacy experiment $\mathrm{Exp}_A^{\mathrm{UFP}}$ involving a $(q, t)$-adversary $A$ is illustrated in Fig. 7. During the learning phase, adversary $A$ chooses a random number $r \in_R [0, q]$, and disturbs $r$ protocol sessions between $\mathcal{R}$ and tag set $\mathcal{T}$ with oracle $(O_i)_{1 \leq i \leq 5}$. Then adversary $A$ outputs useful information $st_0$ and chooses one uncorrupted tag $\mathcal{T}_c$ as its challenge tag. On entering the guessing phase, the experiment chooses a random bit $b$ for adversary $A$, and $b$ is concealed from $A$. Then if $b = 1$, $A$ disturbs $r'$ sessions involving $\mathcal{T}_c$ with oracle $(O_i)_{1 \leq i \leq 4}$. These interactions happen during a single (or several) exposure period of each tag such that $r + r' \leq q$. If $b = 0$, $A$ interacts with random strings rather than true protocol messages in $r'$ protocol session exchanges. Then, $A$ is given the internal state, $st_3$, of $\mathcal{T}_c$ using oracle $O_5$. After this moment, $A$ is no longer able to access any oracle related to $\mathcal{T}_c$, but $A$ can access any other oracle. Then $A$ outputs useful information $st_2$. Eventually, $A$ is asked to guess the random bit $b$ by accessing oracle $(O_i)_{1 \leq i \leq 5}$ to the tag set $\mathcal{T}'$.

Experiment $\mathrm{Exp}_A^{\mathrm{UFP}}(\kappa, N, q, t)$
1. Setup a reader $\mathcal{R}$ and a set of tags $\mathcal{T}$, $|\mathcal{T}| = N$
2. $A$ chooses a random $r \in_R [0, q]$
3. $(\mathcal{T}_c, st_0) \leftarrow A^{(O_i)_{1 \le i \le 5}}(\mathcal{R}, \mathcal{T})$ //learning stage, $r$ sessions
4. Set $\mathcal{T}' = \mathcal{T} - \{\mathcal{T}_c\}$
5. $b \in_R \{0,1\}$ //guessing stage
6. $A$ chooses a random $r'$ such that $r + r' \le q$
7. If $b = 1$, then $st_1 \leftarrow A^{(O_i)_{1 \le i \le 4}}(\mathcal{R}, \mathcal{T}_c)$; otherwise $A$ interacts with random strings and outputs $st_1$ //$r'$ sessions
8. $st_2 \leftarrow A^{O_5}(\mathcal{T}_c)$
9. $b' \leftarrow A(\mathcal{R}, \mathcal{T}', st_0, st_1, st_2)$
10. If $b' = b$ output 1, otherwise output 0

Fig. 7. Unpredictable Forward Privacy experiment

**Definition 8.** The advantage of $(q, t)$-adversary $A$ in the experiment $\mathrm{Exp}_A^{\mathrm{UFP}}$ is defined as:

$$\mathrm{Adv}_A^{\mathrm{UFP}} = \left|\Pr\left[\mathrm{Exp}_A^{\mathrm{UFP}}(\kappa, N, q, t) = 1\right] - \frac{1}{2}\right| \tag{17}$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of the adversary $A$. An authentication protocol is said to be $(q, t, \epsilon)$-forward-private if there exists no $(q, t)$-adversary able to break its unpredictable forward privacy with advantage $\mathrm{Adv}_A^{\mathrm{UFP}} \ge \epsilon$.

This unpredictable forward privacy experiment extends and improves upon the basis of the unpredictable privacy notion proposed by Ha *et al.* (2008). Firstly, the previous model is designed for the general privacy notion in 3-pass and reader initiated protocols, but our experiment has no such limitation, can include any number of passes and protocols initiated by tags. Secondly, the security model presented here uses a variable to simulate the possible transition point between the learning phase and guessing phase. The previous model does not have this property.

**Theorem 4.** Let the oracle $\Pi_{K,\eta}$ in the F-HB⁺ protocol be $(q, t, \epsilon_\Pi)$-secure, let $g$ be a $(t, \epsilon_g)$-secure PRNG, and let $\{h_u: \{0,1\}^l \to \{0,1\}^m\}_{u \subset U}$ be a strongly universal hash function family. Under the attack of a $(q, t)$-adversary, the adversary advantage for the unpredictable forward privacy of the F-HB⁺ protocol can be upper-bounded by

$$\epsilon_{up} = \begin{cases} \epsilon_\Pi + \epsilon_{up\_p}, & \text{successful mutual authentications} \\ \frac{1}{2} + \left(\epsilon_\Pi - \frac{1}{2}\right) P_{FR} + q\epsilon_s + \epsilon_{up\_p}, & \text{otherwise} \end{cases} \tag{18}$$

where $\epsilon_{up\_p} \le (3q + 2)(2q + 1)\epsilon_g + 2Th(m + 3)\left(\epsilon_g + 2^{-l} + 2^{-m} + q2^{-2m+2}\right)$.

Proof. The protocol is composed of an LPN problem and a PRNG, so the forward privacy should be preserved for the LPN problem and PRNG at the same time.

Let us first analyse the forward privacy of the LPN problem. The forward privacy proof of the LPN problem is discussed under two situations. The first situation is that the latest mutual authentication session of the F-HB⁺ protocol before the corruption query in the unpredictable forward privacy experiment is successful. The other one is that the latest session is unsuccessful.

Under the first situation, the tag and the reader can successfully authenticate each other and maintain synchronization. The exchanged messages are random strings and a series of $\Pi_{K,\eta}$ instances, thus, this protocol meets the demands of the unpredictable forward privacy experiment: the exchanged messages cannot be distinguished from random strings. The forward privacy adversary's advantage is upper-bounded by $\epsilon_{\Pi}$ according to Theorem 1.
Under the second situation, the analysis is as follows.

a. If the last tag authentication in the forward privacy experiment is successful, but the adversary uses a desynchronization attack on the reader's acknowledge message, then the reader authentication is unsuccessful. The adversary can obtain the secret and valid LPN instances about this secret, thus he can use this information to check the protocol messages in the previous authentication session. Therefore, the adversary can accurately determine if the previous exchanged messages are random strings.
b. If the last tag authentication in the experiment is unsuccessful, the adversary can obtain the secret and invalid LPN instances about this secret. But these failed instances cannot help him to check the authentication results in previous sessions, because in the LPN problem only the valid instances can help. Therefore, the probability of a correct guess is at most $(1/2 + \epsilon_{\Pi})$ according to Theorem 1.
c. If the adversary can use tag impersonation attacks in the experiment, then the adversary can guess right with probability of 1. The total impersonation probability is at most $q\epsilon_s$.

Therefore, the above situations are combined to illustrate that the forward privacy advantage of the LPN problem is at most

$$\epsilon_{up\_l} \leq (1 - P_{FR}) + \left(\frac{1}{2} + \epsilon_{\Pi}\right) P_{FR} + q\epsilon_s - \frac{1}{2}$$

$$\leq \frac{1}{2} + \left(\epsilon_{\Pi} - \frac{1}{2}\right) P_{FR} + q\epsilon_s. \quad (19)$$

Then, let us discuss the proof of the PRNG. When the authentication is successful, the secret keys of the PRNG cannot be recovered since the key is updated by adding the noise vector. So it is useless to consider the PRNG in this situation. When the authentication is unsuccessful, the secret key of the PRNG is not updated. The possible search length of the PRNG for each session is limited by $Th$, and in each session the PRNG needs to generate $m + 3$ strings (1 for the strong universal hashing, and $m + 2$ for the LPN based MAC).
In the PFP protocol (Berbain *et al.,* 2009), a secure PRNG is used to update the key chain, and a strong universal hash function is used to generate the authentication response. This is similar to the look-up index generation in the F-HB$^+$ protocol. The forward privacy of the PFP protocol can be expressed as in the following Lemma 2.

**Lemma 2** (Berbain *et al.,* 2009). Let $g$ be a $(t, \epsilon_g)$-secure PRNG, let $\{h_u\}_{u \subset U}$ be a strongly universal hash function family, and let $q < \min(2^{m-1}, \omega/2)$ where $\omega$ represents the possible search length of the PRNG. The PFP protocol is $(q, t_p, \epsilon_p)$-forward-private with $\epsilon_p = (3q + 2)(2q + 1)\epsilon_g + 2\omega\left(\epsilon_g + 2^{-l} + 2^{-m} + q2^{-2m+2}\right)$.

Therefore, according to Lemma 2, the forward privacy advantage of the PRNG in the proposed protocol when authentication fails can be expressed as:

$$\epsilon_{up\_p} \leq (3q + 2)(2q + 1)\epsilon_g + 2Th(m + 3)\left(\epsilon_g + 2^{-l} + 2^{-m} + q2^{-2m+2}\right), \quad (20)$$

where $q < \min(2^{m-1}, Th(m+3)/2)$.
Overall, the forward privacy advantage of the proposed protocol can be expressed as:

$$\epsilon_{up} \leq \epsilon_{up\_l} + \epsilon_{up\_p}. \tag{21}$$

Remark. Weak forward privacy in the unsuccessful sessions is as a result of (i) the false rejection probability of the HB related protocols and (ii) desynchronization attacks applied to the reader's acknowledge message in the F-HB+ protocol. However, the false rejection probability $P_{FR}$ can be improved using the parameters proposed by Gilbert *et al.* (2008), and this weak forward privacy is only meaningful to two successive unsuccessful sessions. Therefore, this kind of attack is not very practical.

## 6. Performance evaluation and comparison

### 6.1 Re-Hash collision analysis

In the proposed protocol, an appropriate look-up hash function for the Re-Hash feature must be chosen. The strong universal hash functions can be used due to their excellent collision resistant characteristics. The Toeplitz-based strongly universal hash function is used to analyze the collision performance of hash-table indices after Re-Hash is implemented. According to the random oracle model, the output of a cryptographic hash function can be seen as a random number with uniform distribution. Therefore the inputs to the Re-Hash function have uniform distribution. The collision performance for an output $y \in \{0,1\}^M$ can be measured as follows: how many inputs $x \in \{0,1\}^M$ (as described before, the number of truly usable pseudonyms in each authentication session is equal to the output range) are mapped to the output $y$ by the Re-Hash hash function. Let $S$ be the random variable representing the input number for the same output, then the expected number of $S$ is analyzed as follows:

$$\mathrm{E}[S] = \sum_x \Pr[h_u(x) = y] = 1. \tag{22}$$

The above analysis indicates that the average length in every slot of the hash-table is only 1. Therefore, this hash-table can be used to achieve constant-time performance. After every successful mutual authentication, there are at least Th hash-table slots updated, but the total number of true usable pseudonyms still is kept unchanged, $2^M$. So the above analysis is still valid.

### 6.2 Storage case study

The first case that will be examined is a static system with a fixed tag number. The parameters used by Alomair *et al.* (2010) are adopted to illustrate the practical storage of the proposed protocol. It is assumed that the total number of tags $N$ is $10^9$ and the value of Th is $10^3$. The storage cost of the hash-table is composed of address pointers to the 2nd level database. The storage of pointers is analyzed as follows. The number of elements in the 2nd level is $10^9$ $(= N)$, so the bit-length of a pointer in the 1st level is no more than 30 bits $(\geq \lceil \log_2 N \rceil)$. Therefore, the total storage cost of the hash-table is no more than 4 TB $(\geq N \times Th \times \lceil \log_2 N \rceil)$.
The second case considered is a dynamic system where the tag number can change. Assume the maximum system tag number $N_{MAX}$ is $10^{12}$, and the value of $Th$ is $10^3$. Then the collision-free bit-length of pseudonyms $L$ is 100 bits, and the output range of the Re-Hash

hash function $L'$ is 50 bits. If the initial system tag number $N$ is $10^9$, the initial hash-table slot number $M$ is $10^{12}$. The storage cost can be obtained as follows: (i) the initial table size is upper-bounded to 7 TB ($M \times \lceil \log_2 N_{MAX} \rceil$); (ii) when a new tag is added, $10^3$ slots are added into the dynamic hash-table, and the additional storage is about 7 KB ($Th \times \lceil \log_2 N_{MAX} \rceil$); (iii) when the system number $N$ increases to $N_{MAX}$, the largest table size is no more than 7,000 TB.

### 6.3 Implementation on the tag

Firstly, the PRNG $g(\cdot)$ can be implemented using any candidate in the eSTREAM project (Cid & Robshaw, 2009). If $g(\cdot)$ is implemented using the Grain-v1, only 1,294 gates are required to achieve an 80-bit security level. Secondly, from equations (1) and (6), it can be seen that if the LPN problem is implemented using Toeplitz universal hashing, a linear feedback shift register (LFSR) is required for $T_u$, a 1-bit multiplier plus a 1-bit accumulator is needed for the "$\cdot$" operator, and an XOR operator is also required. Because the $g(\cdot)$ (Grain-v1) needs an LFSR structure, the LPN problem and $g(\cdot)$ can share the LFSR, so $T_u$ can be derived from the state variable of $g(\cdot)$. The two inputs, $x$ and $y$ of the LPN problem can be derived from the output of $g(\cdot)$. Therefore, the main hardware cost of $g(\cdot)$ and the LPN problem equals the hardware cost of $g(\cdot)$ plus a 1-bit "$\cdot$" operator and an XOR. Thus, the final estimate for the hardware cost of these functions is no more than 2,000 gates to achieve an 80-bit security level.

Secondly, the overall hardware cost of the proposed protocol on a tag is 2,000 gates, in addition to the cost of a counter and non-volatile memory for storing the secret key and current value.

### 6.4 Performance comparison

In this section the proposed F-HB$^+$ protocol is compared with previous protocols reported in the literature in terms of their forward privacy properties, the tag resource requirements and the database storage cost. The forward privacy properties are compared in Table 1. Although the proposed protocol cannot protect the forward privacy of failed authentication sessions, it can be observed that it not only supports forward privacy under the unpredictable privacy notion, but also provides a security proof under the standard model.

| | **Le *et al.*, 2007** | **Song, 2009** | **Alomair *et al.*, 2010** | **This work** |
|---|---|---|---|---|
| **Forward Privacy** | For successful sessions | For successful sessions | For successful sessions | For successful sessions |
| **Forward Privacy Notion** | Universal composable notion | Indistinguishable notion | Indistinguishable notion | Unpredictable notion |
| **Forward Privacy Proof** | Universal composable model | Random oracle model | Random oracle model | Standard model |

Table 1. Forward Privacy Comparison Results

The tag hardware cost and desynchronization resistance are compared in Table 2. Although the protocol proposed by Le *et al.* (2007) does not use a counter, it does not provide any

desynchronization resistance because the tag only has one index for a secret key. This work requires only 2,000 gates by using a combination of the LPN problem and a PRNG. And among the three counter-related protocols, the proposed protocol consumes a reasonable non-volatile storage and requires simpler operations in the LPN problem.

| | **Le *et al.*, 2007** | **Song, 2009** | **Alomair *et al.*, 2010** | **This work** |
|---|---|---|---|---|
| **Crypto hardware** | 1 PRF<br>≈ 3,000 gates | 2 $h_C$<br>> 5,000 gates | 1 $h_C$<br>> 5,000 gates | 1 $g$ + 1 LPN<br>≈ 2,000 gates |
| **Non-volatile storage** | 1 key + 1 index | 1 key + 1 $ct_{\mathcal{T}}$ | 2 key + 1 $ct_{\mathcal{T}}$ | 1 key + 1 $ct_{\mathcal{T}}$ |
| **Other hardware** | None | 1 $ct_{\mathcal{T}}$ | 1 $ct_{\mathcal{T}}$ | 1 $ct_{\mathcal{T}}$ |
| **Desynchronization attack resistance** | None | $Th$ | $Th$ | $Th$ |

Table 2. Tag Resource Comparison Results

| | **Le *et al.*, 2007** | **Song, 2009** | **Alomair *et al.*, 2010** | **This work** |
|---|---|---|---|---|
| **Time complexity in synchronization / desynchronization** | $O(1)$ / $O(N)$ | $O(1)$ / $O(N)$ | $O(1)$ / None | $O(1)$ / $O(N)$ |
| **Hash-table storage with the example in (Alomair *et al.*, 2010)** | None | None | 26 TB | 4 TB |
| **Dynamic scalability** | – | – | – | + |

Table 3. Database Performance Comparison Results

The database cost is compared in Table 3. According to the case study for a static system described in section 6.2, the proposed protocol requires storage for the hash-table of no more than 4 TB, but the protocol proposed by Alomair *et al.* (2010) needs about 26 TB. The trade-off in achieving a smaller storage cost is that the proposed protocol needs to compute a look-up table hash function in on-line mode to retrieve the data in the hash-table. The data stored in the hash-table is pre-computed in off-line mode or dynamically inserted in on-line mode. But for the same tag, the look-up procedure and insertion procedure are unlikely to happen at the same time. Because the universal hash function is the fastest hash function in software (Black *et al.*, 1999) and linear hashing is the fastest dynamic hash-table technique, this new look-up hash function will not affect the system performance. Additionally, this proposal is the only to support dynamic scalability.

## 7. Conclusion

In this chapter, the previous authentication protocols for low-cost RFID applications are introduced. In relation to the characteristics of low-cost tags, three important properties are highlighted: (i) hardware cost must be within 200 ~ 3,000 gates, (ii) forward privacy of a tag must be assured, and (iii) scalability of the entire system cannot be compromised.
Therefore, a novel scalable and forward private authentication protocol, F-HB+, is proposed for low-cost RFID tags. The hardware-friendly LPN problem and PRNG are used to reduce

the protocol cost on the tag, which only requires about 2,000 gates plus a hardware counter and some non-volatile memory. A more efficient MAC code is utilized in comparison to the previous F-HB protocol. In the MAC code implementation implementation, a simplified pairwise independent permutation is used to accelerate the MAC code computation, and a PRNG is used to reduce the storage requirement. A new Re-Hash technique is proposed for hash-table based scalable protocols to effectively reduce the storage requirement. In addition, the Re-Hash technique is adapted to a linear-hashing technique, thus, the proposed protocol possesses dynamic scalability. The security proof of the proposed protocol is given under the standard model. It is proven that F-HB+ achieves unpredictable forward privacy for all its transactions before successful mutual authentication sessions.
Finally, a comparison between the proposed protocol and previous protocols is provided. From a hardware perspective, the proposed protocol is among the smallest and it requires the smallest storage cost for its hash-table in addition to supporting dynamic scalability. It also provides unpredictable forward privacy. Overall, the proposed F-HB+ protocol achieves a new and practical balance between hardware cost, scalability and forward privacy.

## 8. References

Avoine, G. (2005). Adversary Model for Radio Frequency Identification, *Technical Report LASEC-REPORT-2005-001*, EPFL, Lausanne, Switzerland, September 2005.

Avoine, G. ; Coisel, I. ; & Martin, T. (2010). Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols. In *Workshop on RFID Security (RFIDSec)*, June 2010.

Alomair, B. ; Clark, A. ; Cuellar, J. ; & Poovendran, R. (2010). Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification. In *IEEE/IFIP International Conference on Dependable Systems and Networks, (DSN'10)*, June 2010.

Black, J. ; Halevi, S. ; Krawczyk, H. ; Krovetz, T. & Rogaway, P. (1999). UMAC: fast and secure message authentication, *Advances in Cryptology – CRYPTO' 99*, LNCS, Volume 1666/1999, 79, DOI: 10.1007/3-540-48405-1_14.

Bringer, J. & Chabanne, H. (2008). Trusted-HB: A Low-Cost Version of HB+ Secure Against Man-in-the-Middle Attacks, *IEEE Transactions on Information Theory* 54(9): 4339-4342 (2008).

Black, P. E. (2009). "linear hashing", in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., *U.S. National Institute of Standards and Technology*. 25 July 2006. Available from: http://xw2k.nist.gov/dads/HTML/linearHashing.html.

Berbain, C. ; Billet, O. ; Etrog J. & Gilbert, H. (2009). An Efficient Forward Private RFID Protocol, *ACM Conference on Computer and Communications Security (CCS)*, November 2009.

Billet, O. ; Etrog, J. & Gilbert, H. (2010). Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher, *International Workshop on Fast Software Encryption (FSE)*, February 2010.

Cid, C. & Robshaw, M. (2009). The eSTREAM Portfolio 2009 Annual Update. July 2009. Available from http://www.ecrypt.eu.org/stream/.

Cao, X & O'Neill, M. (2011). F-HB: An Efficient Forward Private Protocol. *Workshop on Lightweight Security and Privacy: Devices, Protocols and Applications (Lightsec2011)*, March 14-15, 2011, Istanbul, Turkey.

Dimitriou, T. (2005). A Lightweight RFID Protocol to Protect Against Traceability and Cloning attacks. In *International Conference on Security and Privacy in Communication Networks (SecureComm)*, September 2005.

Frumkin, D. & Shamir, A. (2009). Un-Trusted-HB: Security Vulnerabilities of Trusted-HB, *Cryptology ePrint Archive*. Available from : http://eprint.iacr.org/2009/044.

Goldreich, O. (2001). The foundations of Cryptography, Volume I, Basic Tools, *Cambridge University Press*, 2001.

Gilbert, H. ; Robshaw M. J. B. & Seurin, Y. (2008). HB#: Increasing the Security and Efficiency of HB+, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2008: 361-378.

Hopper, N. J. ; & Blum, M. (2001). Secure Human Identification Protocols, *International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT 2001: 52-66.

Henrici, A. & Muller, P. (2004). Hash-based enhancement of location privacy for radiofrequency identification devices using varying identifiers. In R. Sandhu, R. Thomas (Eds.), *International Workshop on Pervasive Computing and Communication Security – PerSec 2004*, IEEE Computer Society, Orlando, Florida, USA, 2004, pp. 149 – 153.

Ha, J. ; Moon, S. ; Zhou J. & Ha, J. (2008). A New Formal Proof Model for RFID Location Privacy, *European Symposium on Research in Computer Security conference (ESORICS)*, October 2008.

Juels, A. & Weis, S. A. (2005). Authenticating Pervasive Devices with Human Protocols, *International Cryptology Conference*, CRYPTO 2005: 293-308.

Juels, A. (2006). RFID Security and Privacy: A research Survey, *IEEE Journal on Selected Areas in Communications*, February 2006.

Juels, A. & Weis, S. (2007). Defining Strong Privacy for RFID, *IEEE Pervasive Computing and Communication (PerCom) conference*, March 2007.

Jr, N.J. *et al.* (2010). Lightweight Cryptographic Algorithms (D.SYM.5) revision 1.0, 1 July 2010. Available from : http://www.ecrypt.eu.org/documents.html.

Krawczyk, H. (1994). LFSR-based hashing and authentication, *International Cryptology Conference, Proc. Crypto'94*, LNCS 839, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 129-139.

Katz, J. & Shin, J. S. (2006). Parallel and Concurrent Security of the HB and HB+ Protocols, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2006: 73-87.

Kiltz, E. ; Pietrzak, K. ; Jain, D. A. & Venturi, D. (2011). Efficient Authentication from Hard Learning Problems. In *Eurocrypt 2011*.

Lim, C. H. & Kwon, T. (2006). Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In *International Conference on Information and Communications Security*, December 2006.

Le, T. V. ; Burmester, M. & de Medeiros, B. (2007). Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange, *ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS)*, March 2007.

Molnar, D. & Wagner, D. (2004). Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *ACM Conference on Computer and Communications Security (CCS)*, October 2004.

Molnar, D. ; Soppera, A. & Wagner, D. (2005). A scalable, delegatable, pseudonym protocol enabling ownership transfer of RFID tags. In *Ecrypt Workshop*, July-August 2005.

Ma, C. ; Li, Y. ; Deng R. & Li, T. (2009). RFID Privacy: Relation Between Two Notions, Minimal Condition, and Efficient Construction, *ACM Conference on Computer and Communications Security (CCS)*, November 2009.

Naor, M. & Reingold, O. (1997). On the Construction of Pseudorandom Permutations: Luby—Rackoff Revisited. In *Journal of Cryptology*, Volume 12, Number 1, 29-66, DOI: 10.1007/PL00003817.

Ohkubo, M. ; Suzuki, K. & Kinoshita, S. (2003). Cryptographic Approach to Privacy-Friendly Tags. *RFID Privacy Workshop*, November 2003.

O'Neill, M. (2008). Low-Cost SHA-1 Hash Function Architecture for RFID Tags. In *RFID Security Workshop 2008 (RFIDSec'08)*, July 2008.

Song, B. (2009). RFID Authentication Protocols using Symmetric Cryptography. In *PhD thesi*s, December 2009. Available from: http://www.avoine.net/rfid/.

Tsudik, G. (2006). YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *IEEE Pervasive Computing and Communication (PerCom) conference*, March 2006.

Vaudenay, S. (2007). On Privacy Models for RFID, *International Conference on the Theory and Application of Cryptology and Information* Security (ASIACRYPT), December 2007.

Wegman, M.N. & Carter, J.L. (1981). New hash functions and their use in authentication and set equality. In *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265-279.

Weis, S. ; Sarma, S. ; Rivest, R. & Engels, D. (2003). Security and privacy aspects of low-cost radio frequency identification systems. In *International Conference on Security in Pervasive Computing*, March 2003.